9. Downloading a File

Write a Ruby script that accept two args the `URL` and the `Path` to download a file from a given URL and saves it locally using `open-uri`, `uri`, `fileutils`.

```
┌──(imenⓀhbtn-lab)-[…/scripting_cyber/0x00-ruby_scripting]
└─$ ls
9-download_file.rb
```

```
┌──(imenⓀhbtn-lab)-[…/scripting_cyber/0x00-ruby_scripting]
└─$ ruby 9-download_file.rb
Usage: 9-download_file.rb URL LOCAL_FILE_PATH
```

```
┌──(imenⓀhbtn-lab)-[…/scripting_cyber/0x00-ruby_scripting]
└─$ ruby 9-download_file.rb
https://logowik.com/content/uploads/images/ruby6530.jpg ./ruby6530.jpg
Downloading file from
https://logowik.com/content/uploads/images/ruby6530.jpg...
File downloaded and saved to ./ruby6530.jpg.
```

```
┌──(imenⓀhbtn-lab)-[…/scripting_cyber/0x00-ruby_scripting]
└─$ls
9-download_file.rb  ruby6530.jpg
```

```ruby
require 'open-uri'
require 'uri'
require 'fileutils'

def download_file(url, local_path)
  # Validate URL
  begin
    uri = URI.parse(url)
    raise "Invalid URL" unless uri.is_a?(URI::HTTP) || uri.is_a?(URI::HTTPS)
  rescue => e
    puts "Error: #{e.message}"
    return
  end

  # Ensure the directory for the local file exists
  FileUtils.mkdir_p(File.dirname(local_path))
```

```ruby
  puts "Downloading file from #{url}..."
  begin
    URI.open(url) do |file|
      File.open(local_path, 'wb') do |local_file|
        local_file.write(file.read)
      end
    end
    puts "File downloaded and saved to #{local_path}."
  rescue => e
    puts "Error downloading file: #{e.message}"
  end
end

# Main script execution
if ARGV.length != 2
  puts "Usage: #{File.basename(__FILE__)} URL LOCAL_FILE_PATH"
else
  url = ARGV[0]
  local_path = ARGV[1]
  download_file(url, local_path)
end
```

How It Works:

1. **Imports Required Modules**:
   - `open-uri`: Enables fetching files from URLs.
   - `uri`: Validates and parses the URL.
   - `fileutils`: Manages the local file system (e.g., creating directories).
2. **Validates the URL**:
   - Uses `URI.parse` to parse and verify the URL.
   - Ensures it is an `http` or `https` URL.
3. **Ensures Directory Exists**:
   - Uses `FileUtils.mkdir_p` to create the directory structure for the specified file path.
4. **Downloads the File**:
   - Opens the URL using `URI.open`.
   - Reads the file's content and writes it to the specified local path using `File.open`.
5. **Handles Errors Gracefully**:
   - Catches and prints errors during validation, downloading, or saving.

6. **Command-Line Usage**:
    - Expects exactly two arguments: the URL and the local file path.
    - Prints a usage message if the arguments are missing.

Example Usage:

1. **Missing Arguments**:

```
$ ruby 9-download_file.rb
Usage: 9-download_file.rb URL LOCAL_FILE_PATH
```

2. **Successful Download**:

```
$ ruby 9-download_file.rb
https://logowik.com/content/uploads/images/ruby6530.jpg ./ruby6530.jpg
Downloading file from
https://logowik.com/content/uploads/images/ruby6530.jpg...
File downloaded and saved to ./ruby6530.jpg.
```

3. **Directory Creation**:

```
$ ruby 9-download_file.rb https://example.com/image.jpg
./downloads/image.jpg
Downloading file from https://example.com/image.jpg...
File downloaded and saved to ./downloads/image.jpg.
```

4. **Invalid URL**:

```
$ ruby 9-download_file.rb invalid-url ./file.jpg
Error: Invalid URL
```

Explanation of Tools:

- **`open-uri`**:
  Fetches the file from the given URL.
- **`File.open`**:
  Writes the fetched content to a local file in binary mode (`wb`).
- **`FileUtils.mkdir_p`**:
  Creates any missing directories in the specified path.

This script should function correctly for downloading and saving files as per the requirements!